A

PROJECT REPORT

FOR

"Disease Prediction through Symptoms Analysis with Machine Learning"

Submitted in partial fulfillment of the requirement for the award of

Bachelor of Technology

In

Computer Science and Engineering

Punyashlok Ahilyadevi Holkar Solapur University

By

Shraddha Annadate	**31**
Kavish Dhirawat	**32**
Sejal Madhekar	**33**
Dnyaneshwari Narale	**34**

Under the Guidance of

Mr. M.A.Mahant

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
WALCHAND INSTITUTE OF TECHNOLOGY
SOLAPUR - 413006

(2022-2023)

Certificate

This is to certify that the project entitled

“Disease Prediction through Symptoms Analysis with Machine Learning”

Is submitted

By

Shraddha Annadate	**31**
Kavish Dhirawat	**32**
Sejal Madhekar	**33**
Dnyaneshwari Narale	**34**

(Mr. M. A. Mahant)
Project Guide

(Dr. Mrs. A.M.Pujar)
Head
Dept of Computer Sc. &Engg

(Dr.V.A.Athaval)

Principal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
WALCHAND INSTITUE OF TECHNOLOGY
SOLAPUR
(2022-2023)

Project Approval Sheet

The Project Entitled

“Disease Prediction through Symptoms Analysis with Machine Learning”

Submitted by

Shraddha Annadate	**31**
Kavish Dhirawat	**32**
Sejal Madhekar	**33**
Dnyaneshwari Narale	**34**

“Is hereby approved in partial fulfillment for the degree of

Bachelor of Computer Science and Engineering”

(Mr. M. A. Mahant)
Project Guide

(Dr. Mrs. A.M.Pujar)
Head
Dept of Computer Sc. &Engg

(Dr.V.A.Athaval)

Principal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
WALCHAND INSTITUTE OF TECHNOLOGY
SOLAPUR - 413006
(2022-2023)

ACKNOWLEDGMENT

At the outset, we would like to take this opportunity to express our deep gratitude to our guide **Mr. M.A. Mahant** of the CSE Department for his guidance and moral support throughout this successful completion of our project.

We heartily thank **Dr. Mrs. A.M. Pujar,** Head of the CSE Dept for her moral support and promoting us through the completion of our project.

We would also like to thank our Principal **Dr. V. A. Athavale** and all staff members for their wholehearted co-operation in completing this project.

UNDERTAKING

We solemnly declare that the project work presented in the report titled "**Disease Prediction through Symptoms Analysis with Machine Learning"** is solely my project work with no significant contribution from any other person except the project guide. Small contribution/help wherever taken has been duly acknowledged and that complete report has been written by the members of the project group.

We understand the zero-tolerance policy of the WIT, Solapur, and the University towards plagiarism. Therefore we as Authors of the above-titled report declare that no portion ofthe report has been plagiarized and any material used as reference is properly referred to / cited.

We undertake that if found guilty of any formal plagiarism in the above-titled report even after the award of the degree, WIT, Solapur and Solapur University reserve the right to withdraw/revoke the degree granted and that WIT, Solapur and the University have the right to publish our name on the website on which names of students are placed who submitted plagiarized report.

Name	Exam Number	University PRN Number	Signature
Shraddha Annadate	194203	2019032500191293	
Kavish Dhirawat	919307	2019032500193075	
Sejal Madekar	194042	2019032500189932	
Dnyaneshwari Narale	214781	2020032500163422	

Date: / /

ABSTRACT

Nowadays, people face various diseases due to environmental condition and their living habits. So the prediction of disease at an earlier stage becomes an important task. But the accurate prediction based on symptoms becomes too difficult for doctors. The correct prediction of disease is the most challenging task. To overcome this problem data mining plays an important role to predict the disease. Medical science has a large amount of data growth per year. Due to the increased amount of data growth in the medical and healthcare field, the accurate analysis of medical data has been beneficial in early patient care.

With the help of disease data, data mining finds hidden pattern information in a huge amount of medical data. We proposed general disease prediction based on the symptoms of the patient. For the disease prediction, we use K- Nearest Neighbor (KNN) and Convolutional neural network (CNN) machine learning algorithm for accurate prediction of disease.

Disease prediction required a disease symptoms dataset. In this general disease prediction, the living habits of the person and checkup information consider for accurate prediction. The accuracy of general disease prediction by using CNN is 84.5% which is more than the KNN algorithm. And thetime and memory requirement is also more in KNN than in CNN. After general disease prediction, this system can give the risk associated with general disease which is a lower risk of general disease or higher.

The dependency on computer-based technology has resulted in the storage of a lot of electronic data in the healthcare industry. As a result of which, health professionals and doctors are dealing with demanding situations to research signs and symptoms correctly and perceive illnesses at an early stage. However, Machine Learning technology has been proven beneficial in giving an immeasurable platform in the medical field so that healthcare issues can be resolved effortlessly and expeditiously.

Disease Prediction is a Machine Learning based system that primarily works according to the symptoms given by a user. The disease is predicted using algorithms and a comparison of the datasets with the symptoms provided by the user.

Index

Chapter 1

INTRODUCTION

Introduction :

Human is expert in understanding information, while the machine is expert at expressing and processing data. we proposed a model for patient symptom similarity analysis by taking advantage of the machine's ability to process data. The model used patients' descriptions of symptoms to extract key information and achieve early prediction and intervention. Therefore, the similarity analysis model's accuracy largely determines the disease prediction's effectiveness. Nowadays, there are many researchers, who have a strong interest in sentence similarity computation and devise some models to compute sentence similarity scores. Socher *et al.* applied recurrent neural networks (RNN) to find and describe images with sentences to test the accuracy of their model. However, the drawbacks of RNN are its gradients vanishing over long sequences and cannot extract dependencies between words. To avoid the vanishing gradients problem, long short-term memory networks (LSTM) were proposed, which are better for learning long-range dependencies. Here we have used LSTM for sentence similarity measurement. The models based on LSTM have excellent performance on tasks of semantic relatedness prediction and sentiment classification.

For the aforementioned problem, we proposed an effective model based on CNN for the symptom similarity analysis. The main contributions of this model can be summarized as follows:

- the raw symptom sentence is not accepted by the model, it should be Since processed and encoded before it is input into the model. Firstly, to reduce the computation burden and improve the model efficiency, we proposed a kernel for extracting the main information about the symptoms of the sentence to preprocess the sentence. Secondly, word embedding technology is applied to map words into vectors to form the vector representation of a patient's symptoms.

- The task of our convolutional neural network is to learn the semantic and syntactic features from the input tensor. Finally, we use the Manhattan distance to compute the score of sentence similarity

- We used different datasets and evaluation criteria to evaluate the model and obtained better results than related work, which ensures the accuracy of patient disease prediction.

Problem Statement and Objective :

The traditional diagnosis approach entails a patient visiting a doctor, undergoing many medical tests, and then reaching a consensus. This process is very time-consuming. This project proposes an automated disease prediction system to save the time required for the initial process of disease prediction that relies on user input.

The user gives input to the system and the system provides the user with a set of probable diseases. There is a need to study and make a system that will make it easy for end users to predict chronic diseases without visiting a physician or doctor for a diagnosis. To detect the Various Diseases through the examining Symptoms of patients using different techniques of Machine Learning Models.

Chapter 2

BACKGROUND

Most of the human diseases are predicted by our system. It accepts the structured type of data as input to the machine learning model. This system is used by end-users i.e. patients/any user. In this system, the user will enter all the symptoms from which he or she is suffering.

The machine learning model will then give these symptoms to predict the disease. Algorithms are then applied to which gives the best accuracy. Then Systemwill predict disease based on symptoms. This system uses Machine Learning Technology.

The Naïve Bayes algorithm is used for predicting the disease by using symptoms, for classification KNN algorithm is used, Logistic regression is used for extracting features which are having most impact value, and the Decision tree is used to divide the big dataset into smaller parts. The final output of this system will be the disease predicted by the model

Chapter3

PROPOSED SOLUTION

Solution :

To implement a robust machine-learning model that can efficiently predict the disease of a human, based on the symptoms that he/she possesses. Let us look into how we can approach this machine-learning problem:

- **Gathering the Data:** Data preparation is the primary step for any machine learning problem. We will be using a dataset from Kaggle for this problem. This dataset consists of two CSV files one for training and one for testing.
- **Cleaning the Data:** Cleaning is the most important step in a machine learning project. The quality of our data determines the quality of our machine-learning model. So it is always necessary to clean the data before feeding it to the model for training. In our dataset all the columns are numerical, the target column i.e. prognosis is a string type and is encoded to numerical form using a label encoder.
- **Model Building:** After gathering and cleaning the data, the data is ready and can be used to train a machine learning model. We will be using this cleaned data to train the Support Vector Classifier, Naive Bayes Classifier, and Random Forest Classifier. We will be using a confusion matrix to determine the quality of the models.
- **Inference:** After training the three models we will be predicting the disease for the input symptoms by combining the predictions of all three models. This makes our overall prediction more robust and accurate.

Reading the dataset

Firstly we will be loading the dataset from the folders using the pandas' library. While reading the dataset we will be dropping the null column. This dataset is a clean dataset with no null values and all the features consist of 0s and 1s. Whenever we are solving a classification task it is necessary to check whether our target column is balanced or not. We will be using a bar plot, to check whether the dataset is balanced or not.

Splitting the data for training and testing the model:

Now that we have cleaned our data by removing the Null values and converting the labels to numerical format, It's time to split the data to train and test the model. We will be splitting the data into 80:20 format i.e. 80% of the dataset will be used for training the model and 20% of the data will be used to evaluate the performance of the models.

Advantages of the proposed system :

The healthcare sector has been the front-runner in adopting digital transformation acrossthe board. Right now, machine learning (ML), a subset of artificial intelligence, is

playing a key role to address health-related areas. This would include having the ability to extract, share & leverage health data and records, development of new medical procedures & even the treatment of chronic diseases. From enhancing operations at a lower cost to improving care quality, ML is revolutionizing every aspect of healthcare with limited human intervention!

And, with the amount of data generated for each patient, ML algorithms in healthcare certainly have great potential. Therefore, it is not surprising that we're witnessing multiple successful ML applications in healthcare right now. In this article, we will explore the role of machine learning in healthcare with its real-world applications and advantages

Chapter 4

WORKING ENVIRONMENT

Hardware Requirements

- System: Any Desktop/Laptop system with the below configuration or higher level.
- Hard disk:500GB
- RAM: 8 GB

Software Requirements

- **Python 3.0 and above** We have some the libraries like Flask.
- Jupyter Notebook: We have these editors, for editing building, and for execution.
- **Kaggle for a dataset:** Kaggle allows users to find datasets they want to use in building AI models, publish datasets, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.
- **Visual Studio:** Visual Studio Code is a streamlined code editor with support fordevelopment operations like debugging, task running, and version control.
- **Github:** Github is a code hosting platform for version control and collaboration.
- **Tkinter:** is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. It's the only framework built into the Python standard library.
- **Sklearn**: Scikit-learn is an open-source data analysis library, and the gold standard for Machine Learning (ML) in the Python ecosystem. Key concepts and features include Algorithmic decision-making methods, including Classification: identifying and categorizing data based on patterns.
- **NumPy**: It can be used to perform a wide variety of mathematical operations on arrays.
- **Matplotlib:** It is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Chapter 5

METHODOLOGY

System Architecture:

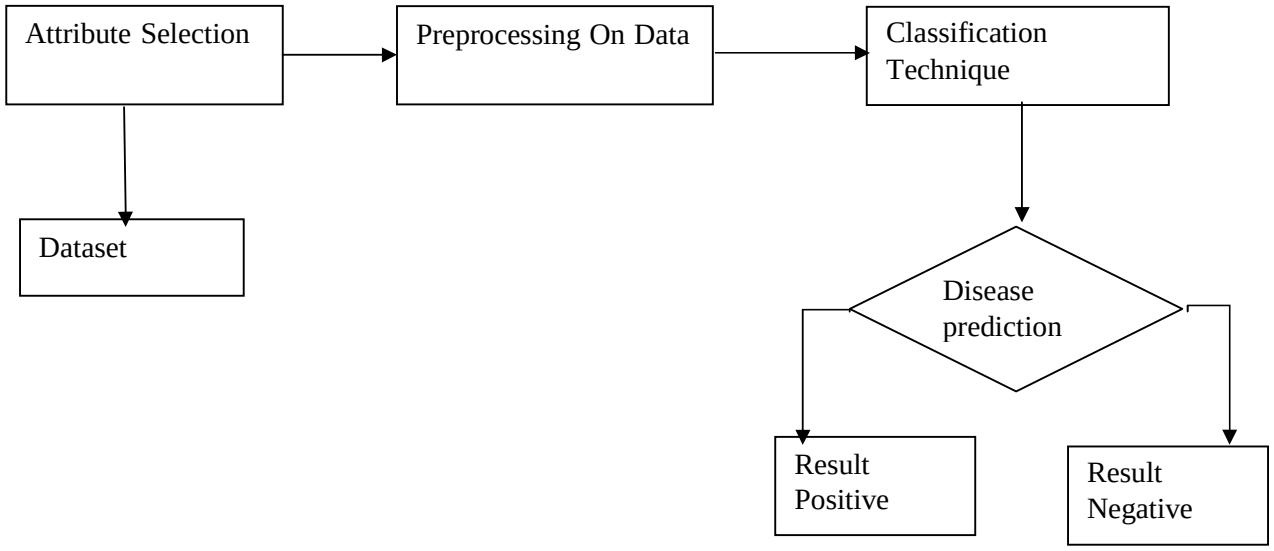


Fig : System architecture

Workflow :

- It is a web-based application that predicts the disease of a user with respect to the symptoms given from different datasets, is carried out, this
- This data will be processed using data trained to predict the dose predict with care application is implemented into two parts, the System and the user part. The duty disease prediction model. The user uses the provided by the model turn and returns the predicted
- If the person has helpful as many people KNN: Suppose there are two categories, point x1, so this data point problem, we need a K-NN to identify the category or class Decision Tree: A decision tree is one of both classification and regression internal node denotes a test and each leaf node (terminal splitting the training data into criterion is met, such as the ma required to split a node.
- It is a web-based application that predicts the disease of the user with respect to the given by the user. The symptom will initially be datasets, the data will be pre-processed before the further process is done so as to get clean data would be noisy, or flowed. will be processed using data mining algorithms, the system will be as to predict the disease based on the input data given by the user. The care application is implemented into two parts, System duty of the system is training the system of the creation model. services provided by the model. The user model after entering the parameter into the model, which predicted the result. any doubt about his/her disease, this system many people have access to the internet 24 hours.

KNN Algorithm :

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning techniques.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for Classification problems.

Steps :

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Decision Tree Classification Algorithm :

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

Naïve Bayes Classifier Algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and

used for solving classification problems.

- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Random Forest Algorithm :

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a

classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Stacking Algorithm :

Stacking is one of the popular ensemble modeling techniques in machine learning. Various weak learners are ensembled in a parallel manner in such a way that by combining them with Meta learners, we can predict better predictions for the future.

This ensemble technique works by applying input of combined multiple weak learners' predictions and Meta learners so that a better output prediction model can be achieved.

In stacking, an algorithm takes the outputs of sub-models as input and attempts to learn how to best combine the input predictions to make a better output prediction.

Stacking is also known as a stacked generalization and is an extended form of the Model Averaging Ensemble technique in which all sub-models equally participate as per their performance weights and build a new model with better predictions. This new model is stacked up on top of the others; this is the reason why it is named stacking.

- **Original data:** This data is divided into n-folds and is also considered test data or training data.

- **Base models:** These models are also referred to as level-0 models. These models use training data and provide compiled predictions (level-0) as an output.
- **Level-0 Predictions:** Each base model is triggered on some training data and provides different predictions, which are known as **level-0 predictions.**
- **Meta Model:** The architecture of the stacking model consists of one meta-model, which helps to best combine the predictions of the base models. The meta-model is also known as the **level-1 model**.
- **Level-1 Prediction:** The meta-model learns how to best combine the predictions of the base models and is trained on different predictions made by individual base models, i.e., data not used to train the base models are fed to the meta-model, predictions are made, and these predictions, along with the expected outputs, provide the input and output pairs of the training dataset used to fit the meta-model.

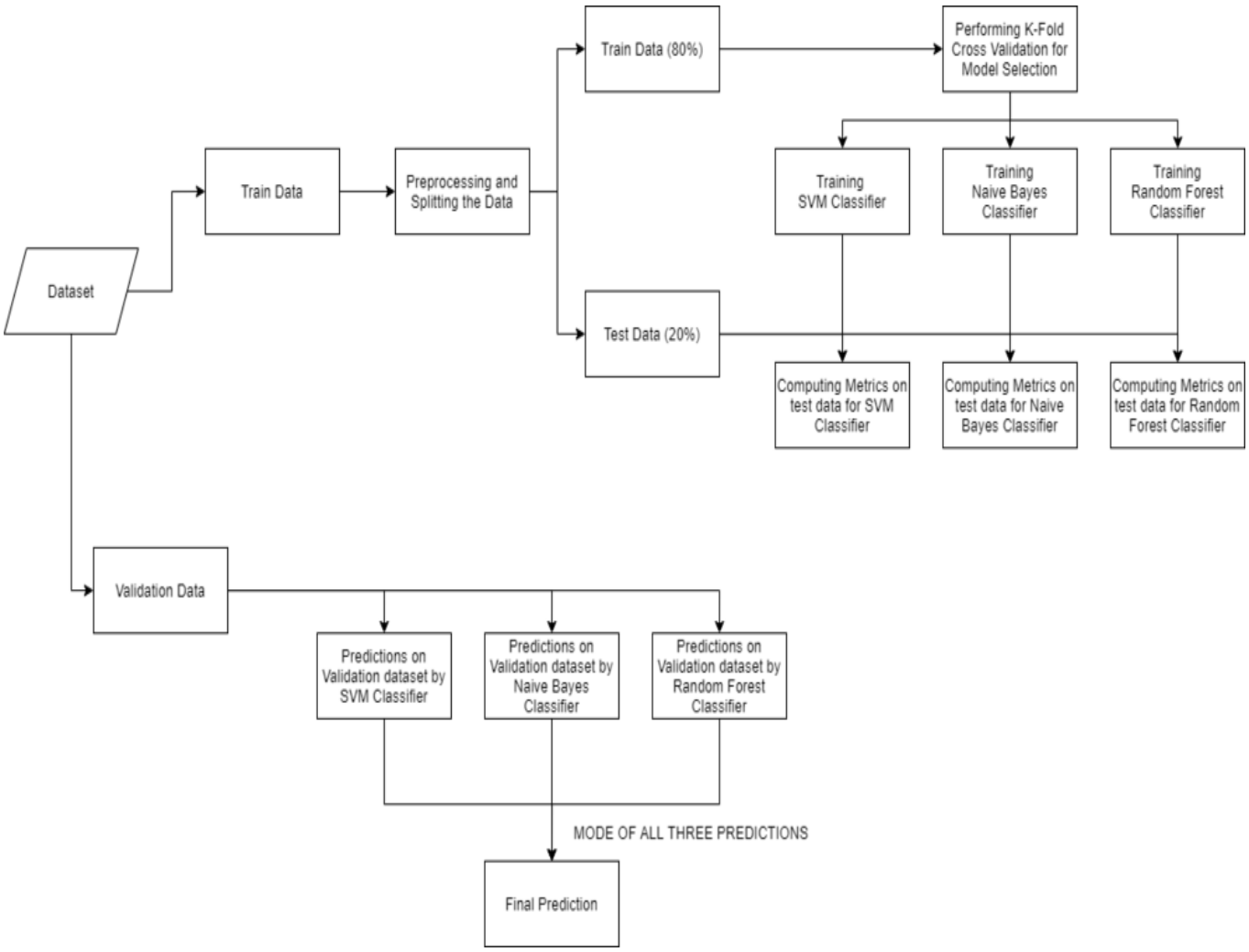
Steps to implement Stacking models:

There are some important steps to implementing stacking models in machine learning. These are as follows:

- Split training data sets into n-folds using the RepeatedStratifiedKFold as this is the most common approach to preparing training datasets for meta-models.
- Now the base model is fitted with the first fold, which is n-1, and it will make predictions for the nth folds.
- The prediction made in the above step is added to the x1_train list.
- Repeat steps 2 & 3 for remaining n-1folds, so it will give x1_train array of size n,
- Now, the model is trained on all the n parts, which will make predictions for the sample data.
- Add this prediction to the y1_test list.
- In the same way, we can find x2_train, y2_test, x3_train, and y3_test by using Model 2 and 3 for training, respectively, to get Level 2 predictions.
- Now train the Meta model on level 1 prediction, where these predictions will be used as features for the model.
- Finally, Meta learners can now be used to make a prediction on test data in the stacking model.

Chapter 6

FLOW DIAGRAMS



Sequential UML Diagrams

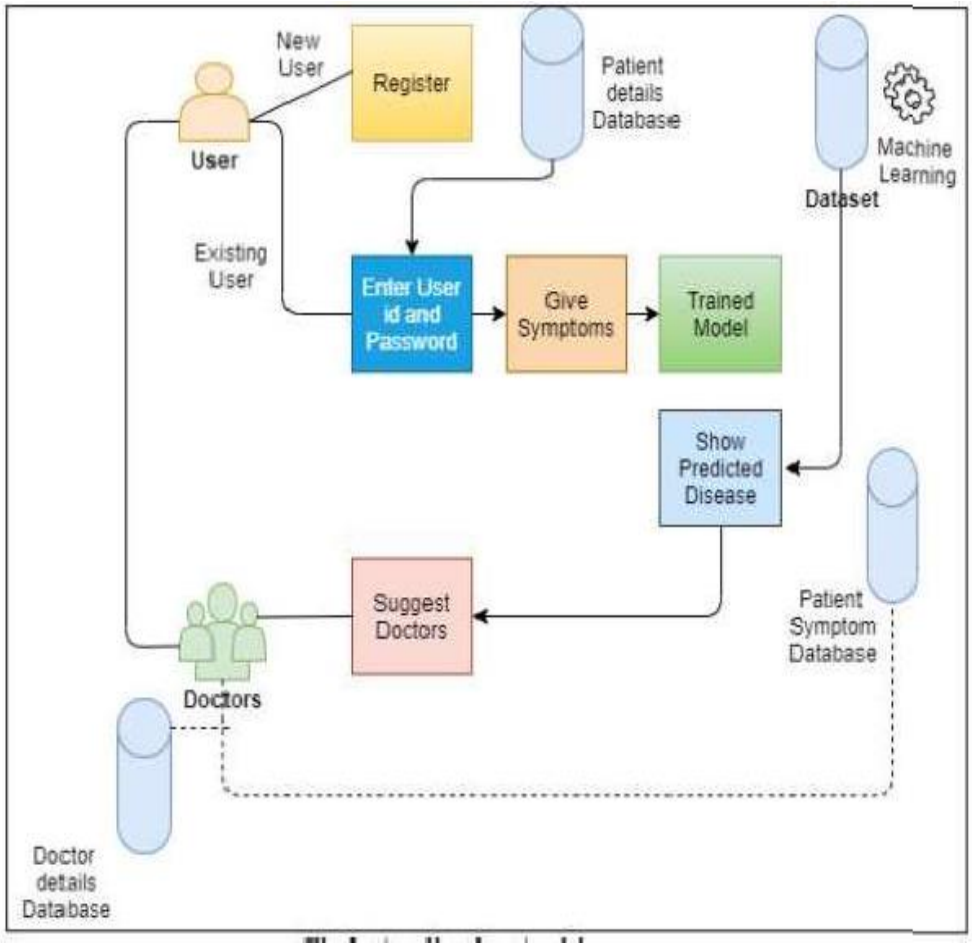


Fig : Sequential UML diagram for proposed System

Chapter 7

#code for the main window

```
import tkinter as tk
import webbrowser
from PIL import ImageTk, Image
from tkinter import ttk


class NavigationBar(tk.Frame):
    def __init__(self, parent, buttons):
        tk.Frame.__init__(self, parent)
        self.buttons = buttons
        self.create_widgets()

    def create_widgets(self):
        # Set the background color of the navigation bar to sky blue
        self.configure(bg="sky blue")

        # Create the buttons without separators
        for button in self.buttons:
            tk.Button(
                self,
                text=button["text"],
                command=button["command"],
                bg="sky blue",  # Set button background color to sky blue
                fg="purple",
                bd=0,
                activebackground="#333",
                activeforeground="#fff",
                padx=20,
                pady=10,
                font=("Times New Roman", 19, "bold"),
            ).pack(side="right", padx=5, pady=5)


if __name__ == "__main__":
    root = tk.Tk()

    # Set the background image
    image_path = 'E:/Project/hal.jpg'
    try:
        image = Image.open(image_path)
        image = image.resize((1920, 1080), Image.ANTIALIAS)
        bg_image = ImageTk.PhotoImage(image)
```

```
        # Create a label with the background image
        bg_label = tk.Label(root, image=bg_image)
        bg_label.place(x=0, y=0, relwidth=1, relheight=1)
        bg_label.image = bg_image  # Store a reference to the image

    except IOError:
        print("Error loading the background image.")

    # Create a placeholder frame for the navigation bar
    nav_frame = tk.Frame(root, bg="sky blue")  # Set frame background color to sky blue
    nav_frame.pack(side="top", fill="x")

    buttons = [
        {"text": "About", "command": show_about},
        {"text": "Search Doctor", "command": open_link},
        {"text": "Tips", "command": show_list},
        {"text": "Test", "command": test},
        {"text": "Disease Prediction", "command": prediction}
    ]
    navigation_bar = NavigationBar(nav_frame, buttons)
    navigation_bar.pack(side="right", padx=10, pady=10)  # Add padding

    # Create a placeholder frame for the title
    title_frame = tk.Frame(root, bg="#222")
    title_frame.pack(side="top", pady=20)

    # Print the title
    title = "Disease Prediction through Symptom Analysis with Machine Learning"
    title_label = tk.Label(title_frame, text=title, font=("Times New Roman", 29, "bold"),
bg="white", fg="black")
    title_label.pack()
    root.geometry("1920x1080")

    root.mainloop()
```

#Disease Prediction code

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from tkinter import *
import numpy as np
from tkinter import Tk
import pandas as pd
```

```
import os
from mpl_toolkits.mplot3d import Axes3D



from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from tkinter import *
from tkinter import messagebox
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
import sqlite3


l1=['back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','yellow_urine',
    'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_stomach',
    'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm','throat_irritation',

'redness_of_eyes','sinus_pressure','runny_nose','congestion','chest_pain','weakness_in_limbs',
    'fast_heart_rate','pain_during_bowel_movements','pain_in_anal_region','bloody_stool',
    'irritation_in_anus','neck_pain','dizziness','cramps','bruising','obesity','swollen_legs',
    'swollen_blood_vessels','puffy_face_and_eyes','enlarged_thyroid','brittle_nails',
    'swollen_extremeties','excessive_hunger','extra_marital_contacts','drying_and_tingling_lips',
    'slurred_speech','knee_pain','hip_joint_pain','muscle_weakness','stiff_neck','swelling_joints',
    'movement_stiffness','spinning_movements','loss_of_balance','unsteadiness',
    'weakness_of_one_body_side','loss_of_smell','bladder_discomfort','foul_smell_of urine',
    'continuous_feel_of_urine','passage_of_gases','internal_itching','toxic_look_(typhos)',
    'depression','irritability','muscle_pain','altered_sensorium','red_spots_over_body','belly_pain',
    'abnormal_menstruation','dischromic
_patches','watering_from_eyes','increased_appetite','polyuria','family_history','mucoid_sputum'
,
    'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_blood_transfusion',
    'receiving_unsterile_injections','coma','stomach_bleeding','distention_of_abdomen',

'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','prominent_veins_on_cal
f',
    'palpitations','painful_walking','pus_filled_pimples','blackheads','scurring','skin_peeling',

'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around_nose
',
    'yellow_crust_ooze']
#List of Diseases is listed in list disease.

disease=['Fungal infection','Allergy','GERD','Chronic cholestasis','Drug Reaction',
    'Peptic ulcer diseae','AIDS','Diabetes','Gastroenteritis','Bronchial Asthma','Hypertension',
```

```
    ' Migraine','Cervical spondylosis',
    'Paralysis (brain hemorrhage)','Jaundice','Malaria','Chicken pox','Dengue','Typhoid','hepatitis
A',
    'Hepatitis B','Hepatitis C','Hepatitis D','Hepatitis E','Alcoholic hepatitis','Tuberculosis',
    'Common Cold','Pneumonia','Dimorphic hemmorhoids(piles)',



'Heartattack','Varicoseveins','Hypothyroidism','Hyperthyroidism','Hypoglycemia','Osteoarthristi
s',
    'Arthritis','(vertigo) Paroymsal  Positional Vertigo','Acne','Urinary tract infection','Psoriasis',
    'Impetigo']
l2=[]
for i in range(0,len(l1)):
    l2.append(0)
print(l2)
df=pd.read_csv("C:\\Users\\SHRIPAD\\Downloads\\Training.csv")

#Replace the values in the imported file by pandas by the inbuilt function replace in pandas.

df.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug
Reaction':4,
    'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial
Asthma':9,'Hypertension ':10,
    'Migraine':11,'Cervical spondylosis':12,
    'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken
pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
    'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic
hepatitis':24,'Tuberculosis':25,
    'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart
attack':29,'Varicose veins':30,'Hypothyroidism':31,
    'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
    '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract
infection':38,'Psoriasis':39,
    'Impetigo':40}},inplace=True)
df.head()
def plotPerColumnDistribution(df1, nGraphShown, nGraphPerRow):
    nunique = df1.nunique()
    df1 = df1[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying
purposes, pick columns that have between 1 and 50 unique values
    nRow, nCol = df1.shape
    columnNames = list(df1)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor
= 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
```

```
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)



  plt.show()
# Scatter and density plots
def plotScatterMatrix(df1, plotSize, textSize):
    df1 = df1.select_dtypes(include =[np.number]) # keep only numerical columns
    # Remove rows and columns that would lead to df being singular
    df1 = df1.dropna('columns')
    df1 = df1[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more
than 1 unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel
density plots
        columnNames = columnNames[:10]
    df1 = df1[columnNames]
    ax = pd.plotting.scatter_matrix(df1, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')
    corrs = df1.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction',
ha='center', va='center', size=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()
#plotPerColumnDistribution(df, 10, 5)
#plotScatterMatrix(df, 20, 10)
X= df[l1]
y = df[["prognosis"]]
np.ravel(y)
print(X)
tr=pd.read_csv("C:\\Users\\SHRIPAD\\Downloads\\Testing.csv")

#Using inbuilt function replace in pandas for replacing the values

tr.replace({'prognosis':{'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug
Reaction':4,
    'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial
Asthma':9,'Hypertension ':10,
    'Migraine':11,'Cervical spondylosis':12,
    'Paralysis (brain hemorrhage)':13,'Jaundice':14,'Malaria':15,'Chicken
pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
    'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic
```

```
hepatitis':24,'Tuberculosis':25,
    'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,'Heart
attack':29,'Varicose veins':30,'Hypothyroidism':31,
    'Hyperthyroidism':32,'Hypoglycemia':33,'Osteoarthristis':34,'Arthritis':35,
    '(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,'Urinary tract
infection':38,'Psoriasis':39,
    'Impetigo':40}},inplace=True)
tr.head()
X_test= tr[l1]
y_test = tr[["prognosis"]]
np.ravel(y_test)



root = Tk()
pred1=StringVar()

def DecisionTree():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn import tree

        clf3 = tree.DecisionTreeClassifier()
        clf3 = clf3.fit(X,y)

        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=clf3.predict(X_test)
        print("Decision Tree")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]

        for k in range(0,len(l1)):
            for z in psymptoms:
```

```
            if(z==l1[k]):
                l2[k]=1

        inputtest = [l2]
        predict = clf3.predict(inputtest)
        predicted=predict[0]

        h='no'
        for a in range(0,len(disease)):
            if(predicted == a):
                h='yes'
                break


        if (h=='yes'):


            pred1.set(" ")
            pred1.set(disease[a])
        else:
            pred1.set(" ")
            pred1.set("Not Found")
        #Creating the database if not exists named as database.db and creating table if not exists
named as DecisionTree using sqlite3
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS DecisionTree(Name StringVar,Symtom1
StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease
StringVar)")
        c.execute("INSERT INTO
DecisionTree(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease)
VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Sy
mptom4.get(),Symptom5.get(),pred1.get()))
        conn.commit()
        c.close()
        conn.close()

        #Random Forest Algorithm
pred2=StringVar()
def randomforest():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
```

```
        if sym:
            root.mainloop()
    else:
        from sklearn.ensemble import RandomForestClassifier
        clf4 = RandomForestClassifier(n_estimators=100)
        clf4 = clf4.fit(X,np.ravel(y))

        # calculating accuracy
        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=clf4.predict(X_test)
        print("Random Forest")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)



        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]

        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1

        inputtest = [l2]
        predict = clf4.predict(inputtest)
        predicted=predict[0]

        h='no'
        for a in range(0,len(disease)):
            if(predicted == a):
                h='yes'
                break
        if (h=='yes'):
            pred2.set(" ")
            pred2.set(disease[a])
        else:
            pred2.set(" ")
            pred2.set("Not Found")
         #Creating the database if not exists named as database.db and creating table if not exists
named as RandomForest using sqlite3
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS RandomForest(Name StringVar,Symtom1
```

```
StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease
StringVar)")
        c.execute("INSERT INTO
RandomForest(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease)
VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Sy
mptom4.get(),Symptom5.get(),pred2.get()))
        conn.commit()
        c.close()
        conn.close()

        #KNearestNeighbour Algorithm
pred4=StringVar()
def KNN():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")


    if sym:
            root.mainloop()
    else:
        from sklearn.neighbors import KNeighborsClassifier
        knn=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
        knn=knn.fit(X,np.ravel(y))

        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=knn.predict(X_test)
        print("KNN")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]

        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1

        inputtest = [l2]
```

```
        predict = knn.predict(inputtest)
        predicted=predict[0]

        h='no'
        for a in range(0,len(disease)):
            if(predicted == a):
                h='yes'
                break


        if (h=='yes'):
            pred4.set(" ")
            pred4.set(disease[a])
        else:
            pred4.set(" ")
            pred4.set("Not Found")
         #Creating the database if not exists named as database.db and creating table if not exists
named as KNearestNeighbour using sqlite3
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS KNearestNeighbour(Name
StringVar,Symtom1 StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4
TEXT,Symtom5 TEXT,Disease StringVar)")


 c.execute("INSERT INTO
KNearestNeighbour(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease)
VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Sy
mptom4.get(),Symptom5.get(),pred4.get()))
        conn.commit()
        c.close()
        conn.close()

        #Naive Bayes Algorithm
pred3=StringVar()
def NaiveBayes():
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp=messagebox.askokcancel("System","Kindly Fill the Name")
        if comp:
            root.mainloop()
    elif((Symptom1.get()=="Select Here") or (Symptom2.get()=="Select Here")):
        pred1.set(" ")
        sym=messagebox.askokcancel("System","Kindly Fill atleast first two Symptoms")
        if sym:
            root.mainloop()
    else:
        from sklearn.naive_bayes import GaussianNB
```

```
        gnb = GaussianNB()
        gnb=gnb.fit(X,np.ravel(y))

        from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
        y_pred=gnb.predict(X_test)
        print("Naive Bayes")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred,normalize=False))
        print("Confusion matrix")
        conf_matrix=confusion_matrix(y_test,y_pred)
        print(conf_matrix)

        psymptoms =
[Symptom1.get(),Symptom2.get(),Symptom3.get(),Symptom4.get(),Symptom5.get()]
        for k in range(0,len(l1)):
            for z in psymptoms:
                if(z==l1[k]):
                    l2[k]=1

        inputtest = [l2]
        predict = gnb.predict(inputtest)
        predicted=predict[0]

        h='no'
        for a in range(0,len(disease)):



       if(predicted == a):
                h='yes'
                break
        if (h=='yes'):
            pred3.set(" ")
            pred3.set(disease[a])
        else:
            pred3.set(" ")
            pred3.set("Not Found")
         #Creating the database if not exists named as database.db and creating table if not exists
named as NaiveBayes using sqlite3
        import sqlite3
        conn = sqlite3.connect('database.db')
        c = conn.cursor()
        c.execute("CREATE TABLE IF NOT EXISTS NaiveBayes(Name StringVar,Symtom1
StringVar,Symtom2 StringVar,Symtom3 StringVar,Symtom4 TEXT,Symtom5 TEXT,Disease
StringVar)")
        c.execute("INSERT INTO
NaiveBayes(Name,Symtom1,Symtom2,Symtom3,Symtom4,Symtom5,Disease)
VALUES(?,?,?,?,?,?,?)",(NameEn.get(),Symptom1.get(),Symptom2.get(),Symptom3.get(),Sy
```

```
mptom4.get(),Symptom5.get(),pred3.get()))
        conn.commit()
        c.close()
        conn.close()

def stacking_model():
    # Check for empty name
    if len(NameEn.get()) == 0:
        pred1.set(" ")
        comp = messagebox.askokcancel("System", "Kindly Fill the Name")
        if comp:
            root.mainloop()
    # Check for at least two symptoms
    elif (Symptom1.get() == "Select Here" or Symptom2.get() == "Select Here"):
        pred1.set(" ")
        sym = messagebox.askokcancel("System", "Kindly Fill at least the first two Symptoms")
        if sym:
            root.mainloop()
    else:
        # Create feature names
        feature_names = ['Symptom1', 'Symptom2', 'Symptom3', 'Symptom4', 'Symptom5']

        # Base models
        clf1 = DecisionTreeClassifier()
        clf2 = RandomForestClassifier(n_estimators=100)
        clf3 = GaussianNB()
        clf4 = KNeighborsClassifier()

        # Meta model
        meta_classifier = LogisticRegression()


# Stacking classifier
        clf_stack = StackingClassifier(
            estimators=[('dt', clf1), ('rf', clf2), ('nb', clf3), ('knn', clf4)],
            final_estimator=meta_classifier
        )

        clf_stack.fit(X, np.ravel(y))

        # calculating accuracy
        from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
        y_pred = clf_stack.predict(X_test)
        print("Stacking Model")
        print("Accuracy")
        print(accuracy_score(y_test, y_pred))
        print(accuracy_score(y_test, y_pred, normalize=False))
        print("Confusion matrix")
        conf_matrix = confusion_matrix(y_test, y_pred)
```

```
    print(conf_matrix)

    psymptoms = [Symptom1.get(), Symptom2.get(), Symptom3.get(), Symptom4.get(),
Symptom5.get()]

    l2 = [0] * len(l1)
    for k in range(0, len(l1)):
        for z in psymptoms:
            if (z == l1[k]):
                l2[k] = 1

    inputtest = [l2]
    predict = clf_stack.predict(inputtest)
    predicted = predict[0]

    h = 'no'
    for a in range(0, len(disease)):
        if (predicted == a):
            h = 'yes'
            break
    if (h == 'yes'):
        pred2.set(" ")
        pred2.set(disease[a])
        result_text.insert(END, disease[a] + "\n")
    else:
        pred2.set(" ")
        pred2.set("Not Found")
        result_text.insert(END, "Not Found\n")

    # Creating the database if not exists named as database.db and creating table if not exists
named as StackingModel
    conn = sqlite3.connect('database.db')
    c = conn.cursor()


 c.execute("CREATE TABLE IF NOT EXISTS StackingModel(Name StringVar,Symptom1
StringVar,Symptom2 StringVar,Symptom3 StringVar,Symptom4 StringVar,Symptom5
StringVar,Disease StringVar)")
    c.execute("INSERT INTO StackingModel (Name, Symptom1, Symptom2, Symptom3,
Symptom4, Symptom5, Disease) VALUES (?, ?, ?, ?, ?, ?, ?)", (
        NameEn.get(), Symptom1.get(), Symptom2.get(), Symptom3.get(), Symptom4.get(),
Symptom5.get(), pred2.get()))
    conn.commit()
    c.close()
    conn.close()
    print(predicted)


root.configure(background='Ivory')
```

```
root.title('Smart Disease Predictor System')
root.geometry("1920x1080")
"
Symptom1 = StringVar()
Symptom1.set("Select Here")

Symptom2 = StringVar()
Symptom2.set("Select Here")

Symptom3 = StringVar()
Symptom3.set("Select Here")

Symptom4 = StringVar()
Symptom4.set("Select Here")

Symptom5 = StringVar()
Symptom5.set("Select Here")
Name = StringVar()
prev_win=None
def Reset():
    global prev_win

    Symptom1.set("Select Here")
    Symptom2.set("Select Here")
    Symptom3.set("Select Here")
    Symptom4.set("Select Here")
    Symptom5.set("Select Here")
    NameEn.delete(first=0,last=100)
    pred1.set(" ")
    pred2.set(" ")
    pred3.set(" ")
    pred4.set(" ")
    try:
        prev_win.destroy()
        prev_win=None
    except AttributeError:


        pass
from tkinter import messagebox
def Exit():
    qExit=messagebox.askyesno("System","Do you want to exit the system")

    if qExit:
        root.destroy()
        exit()
#Headings for the GUI written at the top of GUI
navbar = Frame(root, bg="black")
navbar.pack(side="top", fill="x")
```

```
button = Button(navbar, text="Exit", bg="black", fg="white", font=("Times New Roman", 18,
"bold"), relief="flat")
button.config(width=10, height=2)
button.pack(side="right", padx=7, pady=7)

# Create the text label
label =Label(navbar, text="Disease Prediction through Symptom Analysis with Machine
Learning", font=("Times New Roman", 18, "bold"), bg="black", fg="white")
label.pack(fill="both", expand=True)

#Label for the name
NameLb = Label(root, text="Name of the Patient :", fg="black", bg="Ivory")
NameLb.config(font=("Times",15,"bold"))
NameLb.place(x=400,y=150)
#Creating Labels for the symtoms
S1Lb = Label(root, text="Enter Symptom 1 :", fg="Black", bg="Ivory")
S1Lb.config(font=("Times",15,"bold"))
S1Lb.place(x=400,y=200)

S2Lb = Label(root, text="Enter Symptom 2 :", fg="Black", bg="Ivory")
S2Lb.config(font=("Times",15,"bold"))
S2Lb.place(x=400,y=250)

S3Lb = Label(root, text="Enter Symptom 3 :", fg="Black",bg="Ivory")
S3Lb.config(font=("Times",15,"bold"))
S3Lb.place(x=400,y=300)

S4Lb = Label(root, text="Enter Symptom 4 :", fg="Black", bg="Ivory")
S4Lb.config(font=("Times",15,"bold"))
S4Lb.place(x=400,y=350)

S5Lb = Label(root, text="Enter Symptom 5 :", fg="Black", bg="Ivory")
S5Lb.config(font=("Times",15,"bold"))
S5Lb.place(x=400,y=400)
#Labels for the different algorithms

OPTIONS = sorted(l1)
#Taking name as input from user



NameEn = Entry(root, textvariable=Name)
NameEn.place(x=650,y=150)

#Taking Symptoms as input from the dropdown from the user
S1 = OptionMenu(root, Symptom1,*OPTIONS)
S1.place(x=650,y=200)

S2 = OptionMenu(root, Symptom2,*OPTIONS)
```

```
S2.place(x=650,y=250)

S3 = OptionMenu(root, Symptom3,*OPTIONS)
S3.place(x=650,y=300)

S4 = OptionMenu(root, Symptom4,*OPTIONS)
S4.place(x=650,y=350)

S5 = OptionMenu(root, Symptom5,*OPTIONS)
S5.place(x=650,y=400)
#Buttons for predicting the disease using different algorithms
root.configure(background='skyblue')

def model():
    DecisionTree()
    randomforest()
    NaiveBayes()
    KNN()
    stacking_model()

rs = Button(root,text="Reset Inputs", command=Reset,bg="lightgreen",fg="purple",width=15)
rs.config(font=("Times",15,"bold"))
rs.place(x=650,y=470)


btnModel = Button(root, text="Disease Prediction", command=model, bg="purple",
fg="white", font=("Times", 15, "bold"))
btnModel.place(x=440,y=470)

t4=Label(root,font=("Times",15,"bold"),text="Stacking Model",height=1,bg="pink"
        ,width=40,fg="black",textvariable=pred4,relief="sunken").place(x=410,y=540)


navbar1 = Frame(root, bg="black")
navbar1.pack(side="bottom", fill="x")
button1 = Button(navbar1, text="Test", bg="black", command=test,fg="white", font=("Times
New Roman", 18, "bold"), relief="flat")
button1.config(width=10, height=2)
button1.pack(side="right", padx=7, pady=7)
root.mainloop()
```

#code for test tab

```
def test():

    bg_photo = None
```

```
    def show_list():
        if image_label:
            image_label.pack_forget()

        list_frame = tk.Frame(root, bg="pink")
        list_frame.pack(side="left", fill="y")

        scrollbar = tk.Scrollbar(list_frame)
        scrollbar.pack(side="right", fill="y")

        listbox = tk.Listbox(list_frame, bg="white", fg="black", font=("Times New Roman", 14, "bold"), justify="left", yscrollcommand=scrollbar.set)
        listbox.pack(side="left", fill="both", expand=True)
        scrollbar.config(command=listbox.yview)

        items = ['Fungal infection', 'Allergy', 'GERD', 'Chronic cholestasis',
                 'Drug Reaction', 'Peptic ulcer disease', 'AIDS', 'Diabetes',
                 'Gastroenteritis', 'Bronchial Asthma', 'Hypertension', 'Migraine',
                 'Cervical spondylosis', 'Paralysis (brain hemorrhage)', 'Jaundice',
                 'Malaria', 'Chicken pox', 'Dengue', 'Typhoid', 'Hepatitis A',
                 'Hepatitis B', 'Hepatitis C', 'Hepatitis D', 'Hepatitis E',
                 'Alcoholic hepatitis', 'Tuberculosis', 'Common Cold', 'Pneumonia',
                 'Dimorphic hemorrhoids (piles)', 'Heart attack', 'Varicose veins',
                 'Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia',
                 'Osteoarthristis', 'Arthritis',
                 '(vertigo) Paroxysmal Positional Vertigo', 'Acne',
                 'Urinary tract infection', 'Psoriasis', 'Impetigo']
        for item in items:
            listbox.insert(tk.END, item)

        listbox.bind('<<ListboxSelect>>', show_selected_item)

    def show_selected_item(event):
        selected_item = event.widget.get(event.widget.curselection())
        selected_label.config(text="You Selected Disease: " + selected_item)

        selected_label.place(relx=0.5, rely=0.2, anchor="center")

        if result_button:
            result_button.pack_forget()

        result_button.config(text="Show Result", font=("Times New Roman", 14, "bold"), bg="black", fg="white")

     result_button.place(relx=0.5, rely=0.3, anchor="center")

        global selected_disease
        selected_disease = selected_item
```

```
def show_related_tests():
    tests = {
        'Fungal infection': ['Skin Scraping', 'Blood Tests', 'Serologic Tests','Fungal Culture'],
        'Allergy': ['Skin Prick Test', 'Blood Test', 'Patch Test','Component-Resolved Diagnostic Testing (CRD)'],
        'GERD': ['Esophageal pH monitoring','Esophageal manometry','Barium swallow test','Ambulatory acid (pH) probe test'],
        'Chronic cholestasis':['Liver function tests (LFTs)','Complete blood count (CBC)','Coagulation profile (PT/INR)','Cholesterol levels'],
        'Drug Reaction':['Complete Blood Count (CBC)','Liver Function Tests (LFTs)','Kidney Function Tests (KFTs)','Drug Allergy Testing'],
        'Peptic ulcer disease':['Gastric acid secretion test','Gastrin level test','Complete blood count (CBC)','Fecal occult blood test (FOBT)'],
        'AIDS':['HIV RNA Test','Western Blot Test','PCR (Polymerase Chain Reaction) Test','ELISA (Enzyme-Linked Immunosorbent Assay) Test'],
        'Diabetes':['Fasting Plasma Glucose (FPG) Test','Oral Glucose Tolerance Test (OGTT)','Hemoglobin A1c (HbA1c) Test','Glycated Albumin (GA) Test'],
        'Gastroenteritis':['Stool antigen test','Stool polymerase chain reaction (PCR)','Stool parasite examination','Complete blood count (CBC)'],
        'Bronchial Asthma':['Allergy testing (skin prick test or blood test)','Chest X-ray','High-resolution computed tomography (HRCT) scan of the chest','Bronchial provocation test'],
        'Hypertension':['Blood pressure measurement','Urinalysis','Blood tests (complete blood count, lipid profile, renal function tests)','Electrocardiogram (ECG)'],
        'Migraine':['Blood tests','Imaging tests (such as MRI or CT scan)','Electroencephalogram (EEG)','Visual field test'],
        'Cervical spondylosis':['X-ray of the cervical spine','Magnetic Resonance Imaging (MRI) of the cervical spine','Computed Tomography (CT) scan of the cervical spine','Electromyography (EMG)'],
        'Paralysis (brain hemorrhage)':['Neurological examination','Imaging tests (such as CT scan or MRI)','Electroencephalogram (EEG)','Electromyography (EMG)'],
        'Jaundice':['Liver function tests (LFTs)','Complete blood count (CBC)','Hepatitis virus serology tests','Abdominal ultrasound'],
        'Malaria':['Blood smear microscopy','Rapid diagnostic tests (RDTs)','Polymerase chain reaction (PCR)','Serology tests'],
        'Chicken pox':['Tzanck smear test','Polymerase chain reaction (PCR) test','Serologic tests','Direct fluorescent antibody (DFA) test'],
        'Dengue':['Dengue IgG Antibody Test','Dengue PCR Test','Complete Blood Count (CBC)','Liver Function Tests (LFT)'],
        'Typhoid':['Stool culture','Polymerase chain reaction (PCR) test','Typhidot test','Serology tests'],
        'Hepatitis A':['Hepatitis A IgM Antibody Test','Hepatitis A Total Antibody Test','Hepatitis A RNA Test','Liver Function Tests (LFTs)'],
        'Hepatitis B':['HBsAg (Hepatitis B surface antigen) test','Anti-HBs (Hepatitis B surface antibody)',' testHBcAg (Hepatitis B core antigen) test',' Anti-HBc (Hepatitis B core antibody) test'],
        'Hepatitis C':['HCV RNA test','HCV genotyping test','HCV viral load test','Liver function tests (AST, ALT, bilirubin)'],

'Hepatitis D':['Hepatitis D antigen test','Hepatitis D viral load test','Liver function tests (LFTs)','Complete blood count (CBC)'],
        'Hepatitis E':['Hepatitis E IgG Antibody Test','Hepatitis E RNA Test (PCR)','Hepatitis E Antigen Test','Liver Function Tests (LFTs) - including alanine transaminase (ALT) and aspartate transaminase
```

```
(AST) levels.'],
        'Alcoholic hepatitis':['Complete blood count (CBC)','Serum bilirubin levels','Serum albumin levels','Prothrombin time (PT) or international normalized ratio (INR)'],
        'Tuberculosis':['Tuberculin Skin Test (TST) or Mantoux test','Interferon-Gamma Release Assays (IGRAs) - e.g., QuantiFERON-TB Gold, T-SPOT.TB','Chest X-ray','Sputum Smear Microscopy'],
        'Common Cold':['Throat swab or throat culture','Nasal swab or nasal wash','Rapid antigen test for respiratory viruses','Polymerase chain reaction (PCR) test for respiratory viruses'],
        'Pneumonia':['Chest X-ray','Complete blood count (CBC)','Sputum culture and sensitivity','Blood culture'],
        'Dimorphic hemorrhoids (piles)':['Digital rectal examination','Anoscopy','Proctoscopy','Sigmoidoscopy '],
        'Heart attack':['Electrocardiogram (ECG)','Blood tests (including cardiac enzymes)','Echocardiogram','Stress test (exercise or pharmacological)'],
        'Varicose veins':['Venous duplex ultrasound','Venogram','Magnetic resonance venography (MRV)','CT venography'],
        'Hypothyroidism':['Thyroid-stimulating hormone (TSH) test','Free thyroxine (T4) test','Total triiodothyronine (T3) test','Thyroid peroxidase antibody (TPOAb) test'],
        'Hyperthyroidism':['Thyroid-stimulating hormone (TSH) test','Free thyroxine (FT4) test','Free triiodothyronine (FT3) test','Thyroid peroxidase antibody (TPOAb) test'],
        'Hypoglycemia':['Fasting plasma glucose test','Oral glucose tolerance test (OGTT)','Hemoglobin A1c (HbA1c) test','Insulin levels test'],
        'Osteoarthristis':['Joint fluid analysis','Bone scan','Joint arthroscopy'],
        'Arthritis':['Antinuclear antibody (ANA) test','Erythrocyte sedimentation rate (ESR) test','C-reactive protein (CRP) test','Complete blood count (CBC)'],
        '(vertigo) Paroxysmal Positional Vertigo':['Dix-Hallpike test','Roll test','Head thrust test (also known as head impulse test)','Supine roll test'],
        'Acne':['Bacterial culture','Allergy testing','Skin biopsy','Blood tests'],
         'Urinary tract infection':['Urinalysis','Urine culture and sensitivity test','Complete blood count (CBC)','Kidney function tests'],
        'Psoriasis':['Skin biopsy','Complete blood count (CBC)','Erythrocyte sedimentation rate (ESR)','C-reactive protein (CRP)'],
        'Impetigo':['Bacterial culture','Polymerase chain reaction (PCR)','Antigen detection','Blood tests']
    }

    related_tests = tests.get(selected_disease, [])

    clear_cards()

    for test in related_tests:
        card = tk.Frame(cards_frame, bg="lightblue", padx=10, pady=10)
        card.pack(side="left", padx=10)

        # Upper color gradient
        upper_frame = tk.Frame(card, bg="lightblue", height=40)


        upper_frame.pack(fill="x")
        upper_color_gradient = tk.Canvas(upper_frame, width=200, height=40, bg="lightblue", highlightthickness=0)
```

```
        upper_color_gradient.pack(fill="both", expand=True)
        upper_color_gradient.create_rectangle(0, 0, 200, 40, fill="#f6f6f6", outline="")

        # Test name
        test_label = tk.Label(card, text=test, font=("Times New Roman", 12), fg="purple", bg="lightblue")
        test_label.pack(padx=5, pady=5)

        # Lower color gradient
        lower_frame = tk.Frame(card, bg="lightblue", height=40)
        lower_frame.pack(fill="x")
        lower_color_gradient = tk.Canvas(lower_frame, width=200, height=40, bg="lightblue", highlightthickness=0)
        lower_color_gradient.pack(fill="both", expand=True)
        lower_color_gradient.create_rectangle(0, 0, 200, 40, fill="#d1d1d1", outline="")

    def search_doctor():
        global selected_disease

        if selected_disease:
            search_query = f"nearest doctor for {selected_disease} near me"
            search_url = "https://www.google.com/maps/search/" + search_query
            webbrowser.open(search_url)
    def clear_cards():
        for child in cards_frame.winfo_children():
            child.pack_forget()

    root = tk.Tk()
    root.title("Disease Prediction")
    root.geometry("1920x1080")
    root.resizable(True, True)
    root.config(bg="lightgreen")


    navbar = tk.Frame(root, bg="skyblue")
    navbar.pack(side="top", fill="x")

    button = tk.Button(navbar, text="Diseases", bg="skyblue", fg="purple", font=("Times New Roman", 18, "bold"), relief="flat", command=show_list)
    button.config(width=10, height=2)
    button.pack(side="left", padx=7, pady=7)

    label = tk.Label(navbar, text="Disease Prediction through Symptom Analysis with Machine Learning", font=("Times New Roman", 22, "bold"), bg="skyblue", fg="purple")
    label.pack(fill="both", expand=True)


    selected_label = tk.Label(root, text="", font=("Times New Roman", 14, "bold"), fg="purple", bg="white")
```

```
    result_button = tk.Button(root, text="Get Result", font=("Times New Roman", 14, "bold"), bg="purple", fg="white", command=show_related_tests)

    cards_frame = tk.Frame(root, bg="white")

    image_label = tk.Label(root)

    cards_frame.place(relx=0.5, rely=0.5, anchor="center")

    image_label.pack(side="left")


    nav_frame = tk.Frame(root, bg="black")
    nav_frame.pack(side=tk.BOTTOM, fill=tk.X)

    exit_button = tk.Button(nav_frame, text="Exit", bg="black", fg="white", font=("Times New Roman", 16, "bold"),
                            relief="flat", command=root.destroy)
    exit_button.pack(side="right", padx=10, pady=10)


    doctor = tk.Button(nav_frame, text="Search Doctor", bg="black", fg="white", font=("Times New Roman", 16, "bold"),
                            relief="flat", command=search_doctor)
    doctor.pack(side="right", padx=10, pady=10)
    selected_label.pack()
    result_button.pack()

    root.mainloop()
test()
```

#code for tips tab

```
import tkinter as tk
import requests
from bs4 import BeautifulSoup
import webbrowser
import re

# Declare global variables
selected_disease = None
cards_frame = None


def show_list():
```

```
    global cards_frame

    if cards_frame:
        clear_cards()

    items = ['Fungal infection', 'Allergy', 'GERD', 'Chronic cholestasis',
             'Drug Reaction', 'Peptic ulcer disease', 'AIDS', 'Diabetes',
             'Gastroenteritis', 'Bronchial Asthma', 'Hypertension', 'Migraine',
             'Cervical spondylosis', 'Paralysis (brain hemorrhage)', 'Jaundice',
             'Malaria', 'Chicken pox', 'Dengue', 'Typhoid', 'Hepatitis A',
             'Hepatitis B', 'Hepatitis C', 'Hepatitis D', 'Hepatitis E',
             'Alcoholic hepatitis', 'Tuberculosis', 'Common Cold', 'Pneumonia',
             'Dimorphic hemorrhoids (piles)', 'Heart attack', 'Varicose veins',
             'Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia',
             'Osteoarthristis', 'Arthritis',
             '(vertigo) Paroxysmal Positional Vertigo', 'Acne',
             'Urinary tract infection', 'Psoriasis', 'Impetigo']

    disease_list = tk.Listbox(root, bg="white", fg="black", font=("Times New Roman", 16, "bold"),
justify="left", width=5)
    disease_list.pack(side="left", fill="both", expand=True)

    for item in items:
        disease_list.insert(tk.END, item)

    disease_list.bind('<<ListboxSelect>>', show_selected_item)


def show_selected_item(event):
    global selected_disease, cards_frame

    selected_item = event.widget.get(event.widget.curselection())
    selected_disease = selected_item

    related_tips = get_related_tips(selected_disease)

    if cards_frame:
        cards_frame.destroy()

    cards_frame = tk.Frame(root, bg="white")
    cards_frame.pack(side="left", padx=10, pady=10, fill="both", expand=True)

    scrollbar = tk.Scrollbar(cards_frame)
    scrollbar.pack(side="right", fill="y")

    canvas = tk.Canvas(cards_frame, bg="white", yscrollcommand=scrollbar.set)
    canvas.pack(side="left", fill="both", expand=True)
```

```
    scrollbar.config(command=canvas.yview)

    inner_frame = tk.Frame(canvas, bg="white")
    canvas.create_window((0, 0), window=inner_frame, anchor="nw")

    num_columns = 2  # Number of columns for the card layout
    max_cards = 4  # Maximum number of cards to display

    # Counter for the number of cards created
    card_counter = 0

    all_tips = ""

    for i, tip in enumerate(related_tips):
        if card_counter >= max_cards:
            break

        # Skip tips that contain the date
        if any(month in tip for month in ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov',
'Dec')):
            continue

        card_counter += 1

        # Add tip to the string
        all_tips += tip + "\n\n"

    # Create a single card for all the tips
    card = tk.Frame(inner_frame, bg="lightblue", padx=10, pady=10)
    card.pack(padx=10, pady=10)

    # Disease label
    disease_label = tk.Label(card, text=selected_disease, font=("Arial", 16, "bold"), fg="purple",
bg="lightblue")
    disease_label.pack(padx=5, pady=5)

    # Tip label
    tip_label = tk.Label(card, text=all_tips, font=("Arial", 12), fg="purple", bg="lightblue",
wraplength=400,
                         justify="left")
    tip_label.pack(padx=5, pady=5)

    # Web search button
    def search_web():
        search_query = selected_disease + " do's and dont's"
        search_url = "https://www.google.com/search?q=" + search_query
        webbrowser.open(search_url)
```

```
    web_button = tk.Button(inner_frame, text="Search on the Web", bg="black", fg="white",
                           font=("Arial", 12), command=search_web)
    web_button.pack(pady=5)

    # YouTube button
    def search_youtube():
        search_query = selected_disease + " videos"
        search_url = "https://www.youtube.com/results?search_query=" + search_query
        webbrowser.open(search_url)

    youtube_button = tk.Button(inner_frame, text="Search on YouTube", bg="black", fg="white",
                               font=("Arial", 12), command=search_youtube)
    youtube_button.pack(pady=5)

    def update_canvas_scrollregion(event):
        canvas.configure(scrollregion=canvas.bbox("all"))

    inner_frame.bind("<Configure>", update_canvas_scrollregion)


def clear_cards():
    global cards_frame
    if cards_frame:
        cards_frame.destroy()
    cards_frame = tk.Frame(root, bg="white")


def get_related_tips(disease):
    search_query = disease + " tips"
    search_url = "https://www.google.com/search?q=" + search_query

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0;Win64) AppleWebkit/537.36 (KHTML, like
Gecko) Chrome/58.0.3029.110 Safari/537.3"}
    response = requests.get(search_url, headers=headers)
    soup = BeautifulSoup(response.content, "html.parser")
    search_results = soup.find_all("div", {"class": "BNeawe s3v9rd AP7Wnd"})
    tips = [result.get_text(strip=True) for result in search_results]

    # Filter out incorrect tips
    filtered_tips = [tip for tip in tips if re.match(r"^[A-Z].*[\.\?!]$", tip)]

    return filtered_tips


def search_doctor():
    global selected_disease
```

```
    if selected_disease:
        search_query = f"nearest doctor for {selected_disease} near me"
        search_url = "https://www.google.com/maps/search/" + search_query
        webbrowser.open(search_url)


root = tk.Tk()
root.title("Disease Prediction")
root.geometry("800x800")
root.resizable(True, True)

navbar = tk.Frame(root, bg="purple")
navbar.pack(side="top", fill="x")

button = tk.Button(navbar, text="Diseases", bg="Purple", fg="white", font=("Times New Roman", 18,
"bold"),
                   relief="flat", command=show_list)
button.config(width=10, height=2)
button.pack(side="left", padx=10, pady=10)


title_label = tk.Label(navbar, text="Disease Prediction through Symptom Analysis with Machine
Learning", bg="purple", fg="white",
                       font=("Times New Roman", 18, "bold"), relief="flat")
title_label.pack(side="left", padx=10, pady=10)

selected_label = tk.Label(root, text="", bg="white", font=("Arial", 18))

cards_frame = tk.Frame(root, bg="white")
cards_frame.pack(side="left", padx=10, pady=10, fill="both", expand=True)

# Lower section
lower_frame = tk.Frame(root, bg="black")
lower_frame.pack(side="bottom", fill="x")

exit_button = tk.Button(lower_frame, text="Exit", bg="black", fg="white", font=("Times", 12, "bold"),
                        relief="flat", command=root.destroy)
exit_button.pack(side="right", padx=10, pady=10)

search_doctor_button = tk.Button(lower_frame, text="Search Doctor", bg="black", fg="white",
font=("Times New Roman", 12, "bold"),
                                 relief="flat", command=search_doctor)
search_doctor_button.config(width=15, height=2)
search_doctor_button.pack(side="right", padx=10, pady=10)

root.mainloop()
```

#code for search nearest doctor

```
def search_doctor():
    global selected_disease

    if selected_disease:
        search_query = f"nearest doctor for {selected_disease} near me"
        search_url = "https://www.google.com/maps/search/" + search_query
        webbrowser.open(search_url)
```

#code for about tab

```
def show_about():
    about_window = tk.Toplevel(root)
    about_window.title("About")
    about_window.geometry("780x800")
    about_window.configure(bg="orange")  # Set background color of the second window

    # Create a navigation bar
    navigation_frame = tk.Frame(about_window, bg="black", height=100)
    navigation_frame.pack(side="top", fill="x")

    # Add text to the navigation bar
    navigation_label = tk.Label(navigation_frame, text="Disease Prediction through Symptom Analysis with Machine Learning", font=("Times New Roman", 16, "bold"), bg="black", fg="white")
    navigation_label.pack(side="left", padx=20, pady=20, fill="both", expand=True)  # Use fill and expand parameters to center the text

    # Create navigation buttons
    about_button = tk.Button(navigation_frame, bg="black", fg="white", text="Exit", relief="flat", command=about_window.destroy, font=("Times New Roman", 16,"bold"))
    about_button.pack(side="right", padx=20, pady=20)

    # Create a canvas to hold the content
    canvas = tk.Canvas(about_window, bg="white")
    canvas.pack(fill="both", expand=True)

    # Add the background image
    background_image = ImageTk.PhotoImage(Image.open("E:/Project/diagnose.jpg"))
    canvas.create_image(0, 0, anchor="nw", image=background_image)

    # Create a vertical scrollbar
    scrollbar_y = ttk.Scrollbar(about_window, orient="vertical", command=canvas.yview)
    scrollbar_y.pack(side="right", fill="y")
```

```
    # Create a horizontal scrollbar
    scrollbar_x = ttk.Scrollbar(about_window, orient="horizontal", command=canvas.xview)


     scrollbar_x.pack(side="bottom", fill="x")

    # Configure the canvas to use the scrollbars
    canvas.configure(yscrollcommand=scrollbar_y.set, xscrollcommand=scrollbar_x.set)

    # Add the content frame
    content_frame = tk.Frame(canvas, bg="white")
    canvas.create_window((0, 0), window=content_frame, anchor="nw")

    # Add the content to the frame
    add_content(content_frame)

    # Update the canvas scroll region
    content_frame.update_idletasks()
    canvas.configure(scrollregion=canvas.bbox("all"))

def add_content(frame):
    # Create a row frame for the disease prediction, test, and diagnose tabs
    row_frame_1 = tk.Frame(frame, bg="white")
    row_frame_1.pack(pady=20)

    # Add disease prediction tab
    add_card(row_frame_1, "E:/Project/diagnose.jpg", "Disease Prediction Tab",
             """In this tab, the user selects five health parameters (possibly symptoms or other indicators), and based on these inputs, the system predicts the most likely disease. It's not clear from the code how the prediction is made or what algorithms are used.""")

    # Add test tab
    add_card(row_frame_1, "E:/Project/test.jpg", "Test Tab",
             """After the disease is predicted in the Disease Prediction Tab, this tab provides the names of tests that are typically associated with the predicted disease. These tests can help confirm or rule out the predicted disease. The tests are displayed in a list format.""")

    # Add diagnose tab
   # add_card(row_frame_1, "E:/Project/diagnose.jpg", "Diagnose Tab",

    #"""In this tab, the system performs disease prediction based on health parameters. It's not specified in the code how these health parameters are obtained or what algorithms are used for diagnosis. It seems to be a separate functionality from the Disease Prediction Tab, but the details are not provided.""") # Create a row frame for the tips and nearest doctor tabs

    row_frame_2 = tk.Frame(frame, bg="white")
    row_frame_2.pack(pady=20)

    # Add tips tab
```

```
    add_card(row_frame_2, "E:/Project/diet.jpg", "Tips Tab",
             """This tab provides tips related to the predicted disease. The tips can include suggestions for
```

yoga exercises, dietary recommendations, and links to relevant YouTube videos. However, the code you provided doesn't contain the implementation of this tab, so it's not clear how the tips are generated or displayed.""")

```
    # Add nearest doctor tab
    add_card(row_frame_2, "E:/Project/log.png", "Nearest Doctor Tab",
             """This tab helps users find the nearest doctors or medical professionals based on their location
```

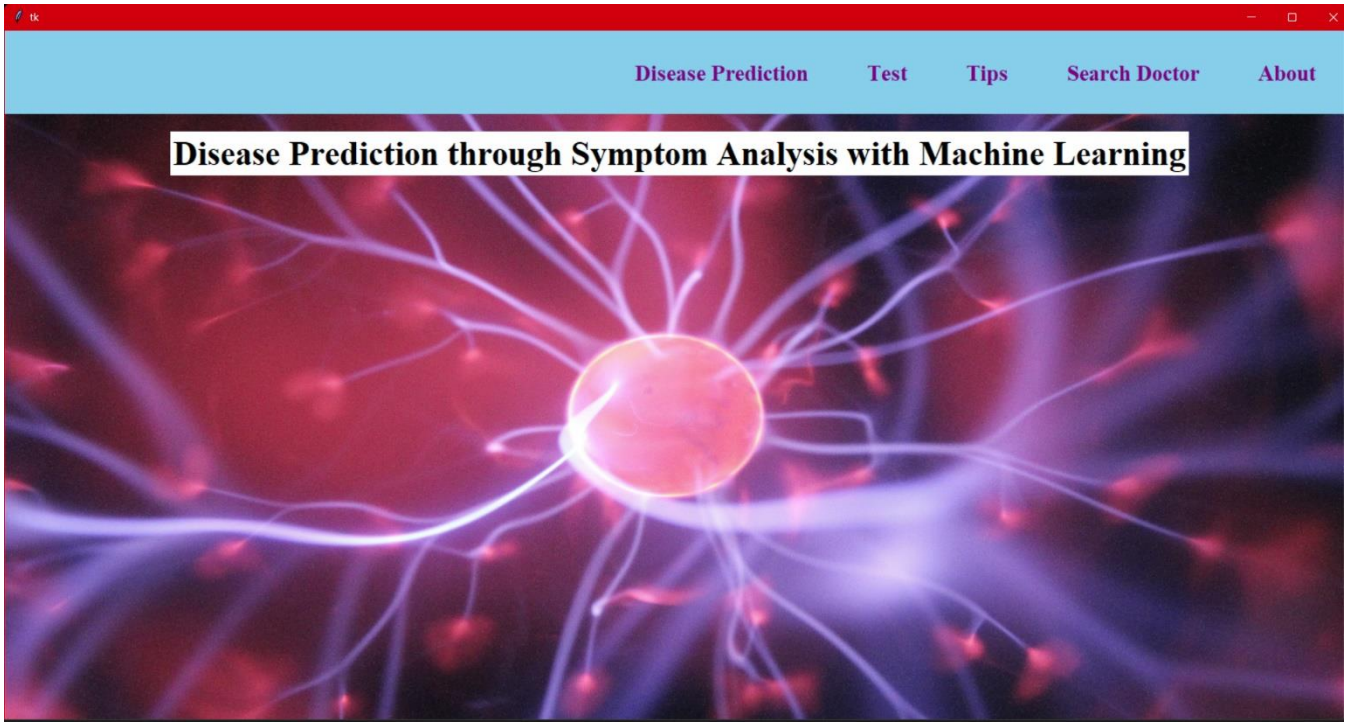
or other search criteria. The code for this tab is not provided, so the implementation details are unknown.""")

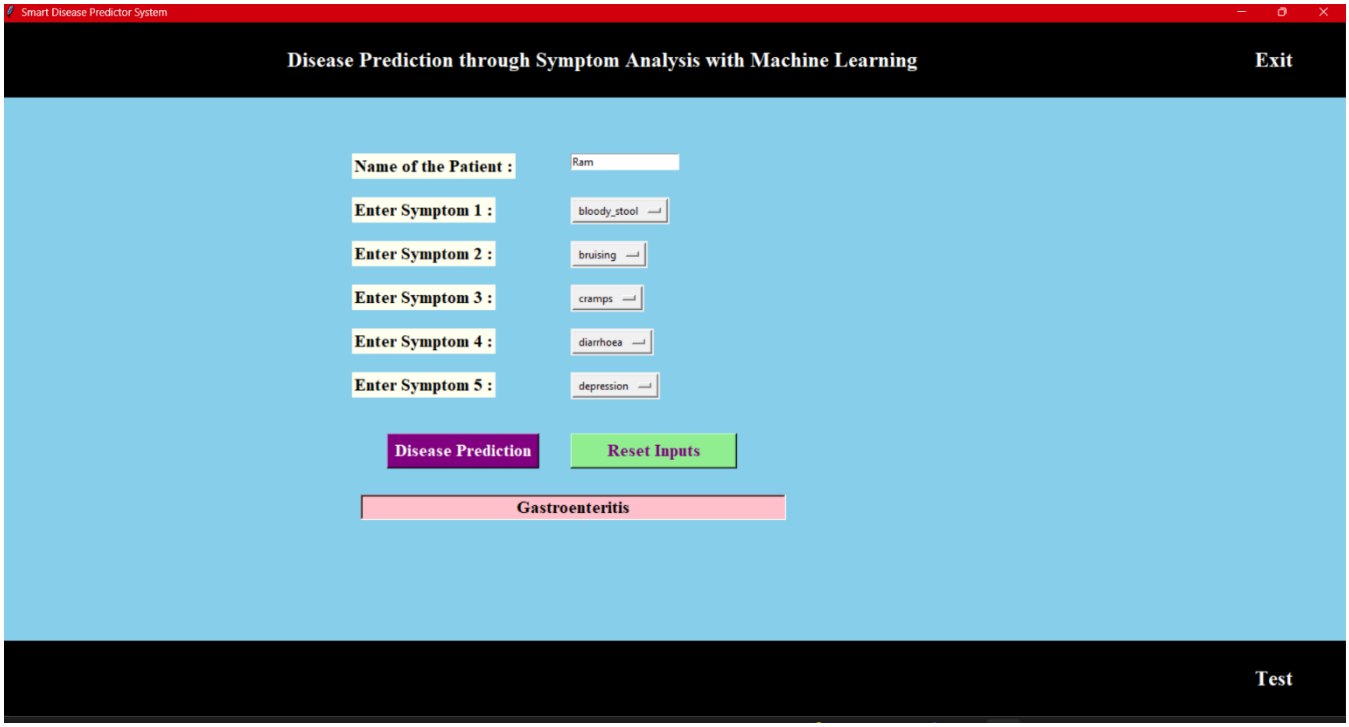
```
def add_card(frame, image_path, title, text):
    # Create a card frame
    card_frame = tk.Frame(frame, bg="lightblue", width=400, height=300, padx=10, pady=10)
    card_frame.pack(side="left", padx=20)

    # Add an image to the card
    image = Image.open(image_path)
    image = image.resize((100, 100))  # Adjust the size as needed
    photo = ImageTk.PhotoImage(image)
    image_label = tk.Label(card_frame, image=photo, bg="lightblue")
    image_label.image = photo
    image_label.pack()

    # Add the title and text to the card
    title_label = tk.Label(card_frame, text=title, font=("Times New Roman", 16, "bold"), bg="lightblue")
    title_label.pack()

    text_label = tk.Label(card_frame, text=text, font=("Times New Roman", 12), bg="lightblue",
```

wraplength=300, justify="left")

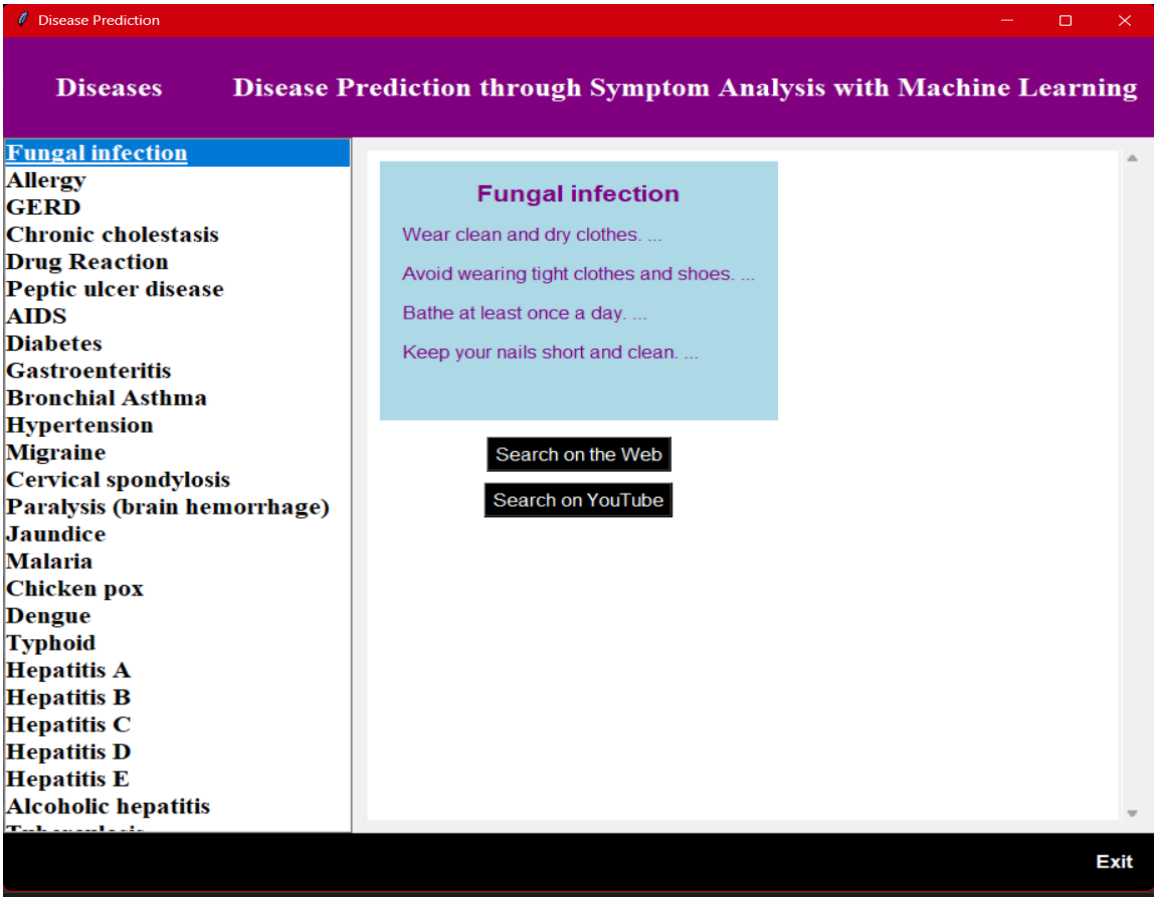
```
    text_label.pack()

root.mainloop()
show_about()
```

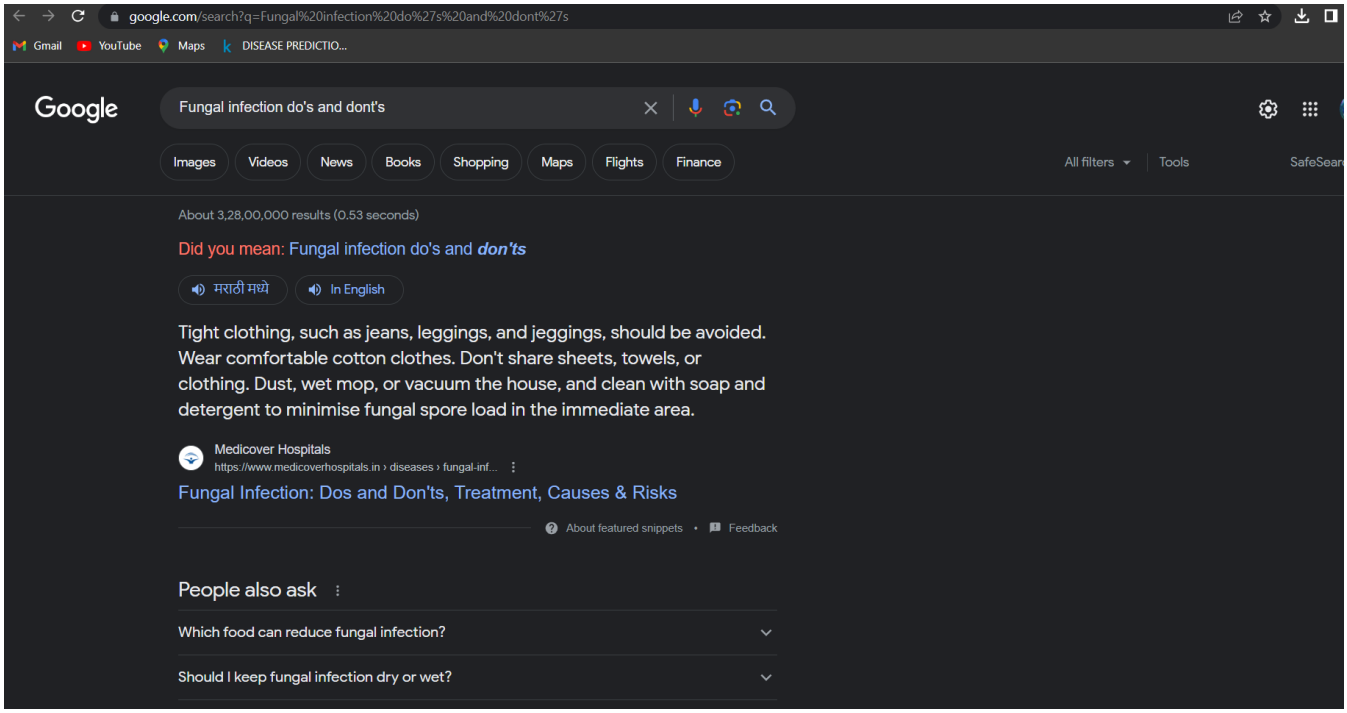
Screenshots and Results :

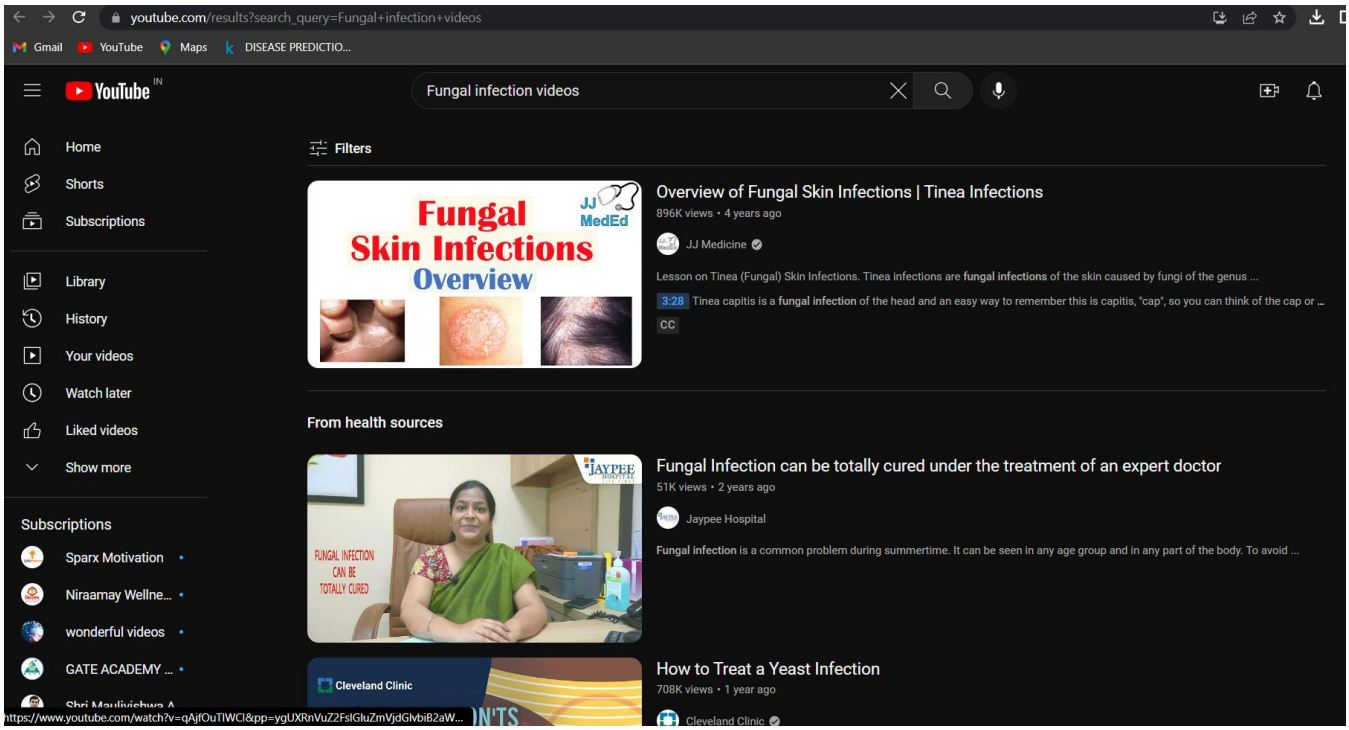


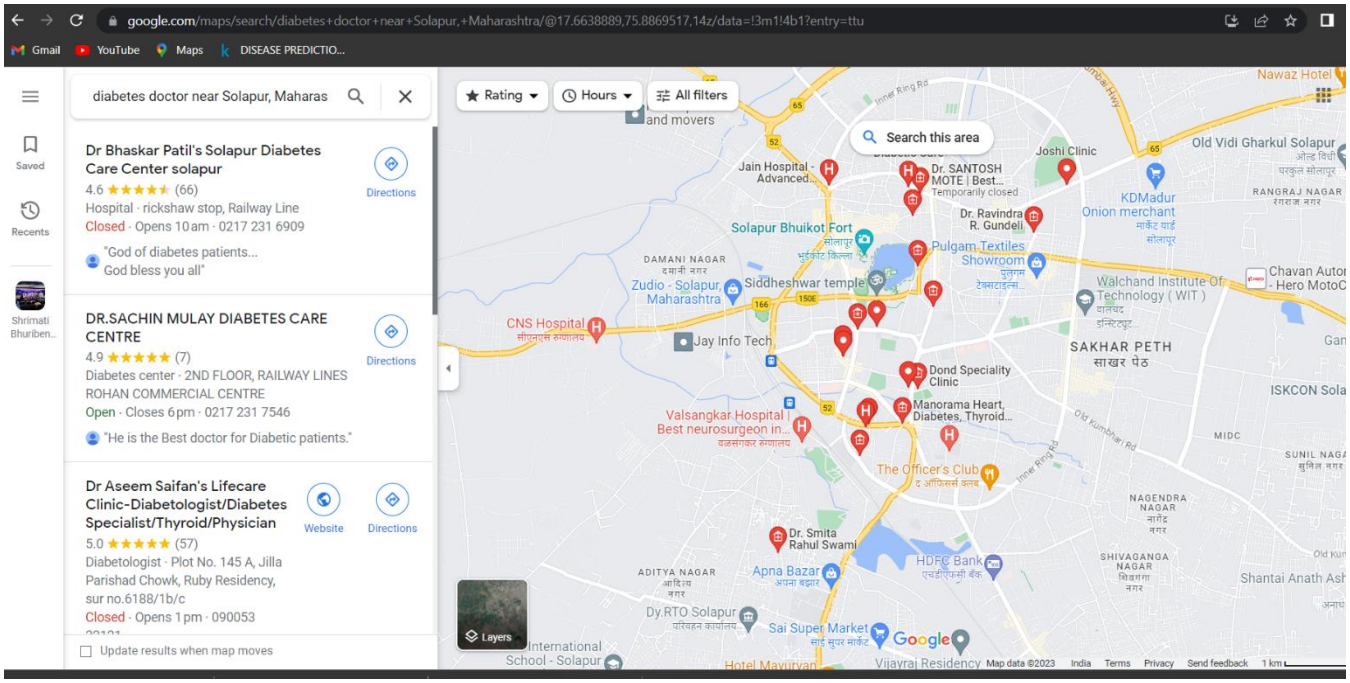


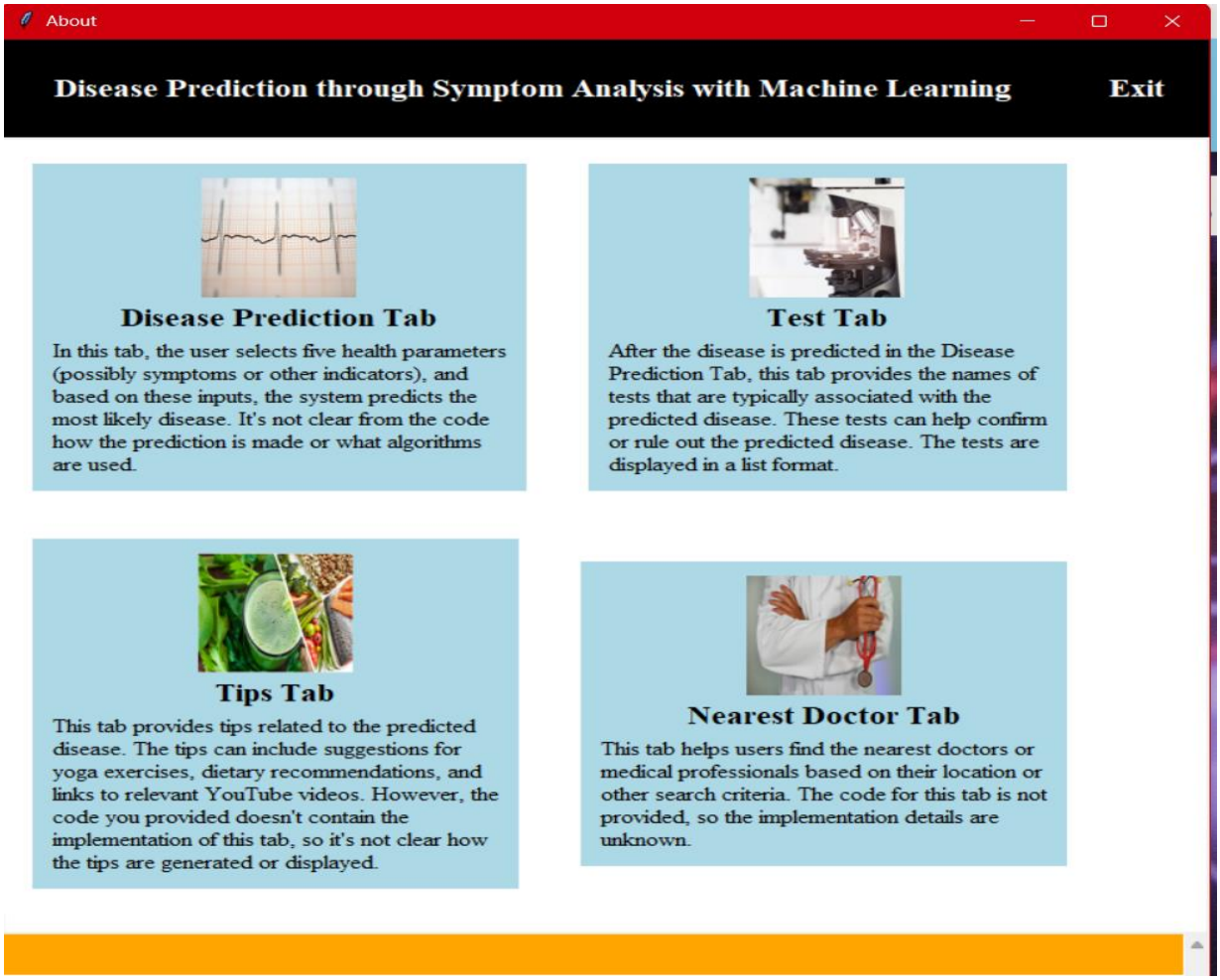

Disease Prediction
Diseases
Disease Prediction through Symptom Analysis with Machine Learning
Fungal infection
Allergy
GERD
Chronic cholestasis
Drug Reaction
Peptic ulcer disease
AIDS
Diabetes
Gastroenteritis
Bronchial Asthma
Hypertension
Migraine
Cervical spondylosis
Paralysis (brain hemorrl
Jaundice
Malaria
Chicken pox
Dengue
Typhoid
Hepatitis A
Hepatitis B
Hepatitis C
Hepatitis D
Hepatitis E
Alcoholic hepatitis
Tuberculosis
Common Cold
Pneumonia
Dimorphic hemorrhoids
You Selected Disease: Fungal infection
Show Result
Skin Scraping
Blood Tests
Serologic Tests
Fungal Culture


Disease Prediction
Diseases
Disease Prediction through Symptom Analysis with Machine Learning
Fungal infection
Allergy
GERD
Chronic cholestasis
Drug Reaction
Peptic ulcer disease
AIDS
Diabetes
Gastroenteritis
Bronchial Asthma
Hypertension
Migraine
Cervical spondylosis
Paralysis (brain hemorrhage)
Jaundice
Malaria
Chicken pox
Dengue
Typhoid
Hepatitis A
Hepatitis B
Hepatitis C
Hepatitis D
Hepatitis E
Alcoholic hepatitis
Fungal infection
Wear clean and dry clothes. ...
Avoid wearing tight clothes and shoes. ...
Bathe at least once a day. ...
Keep your nails short and clean. ...
Search on the Web
Search on YouTube
Exit


google.com/search?q=Fungal%20infection%20do%27s%20and%20dont%27s
Gmail
YouTube
Maps
DISEASE PREDICTIO...
Google
Fungal infection do's and dont's
Images
Videos
News
Books
Shopping
Maps
Flights
Finance
All filters
Tools
About 3,28,00,000 results (0.53 seconds)
Did you mean: Fungal infection do's and don'ts
मराठी मध्ये
In English
Tight clothing, such as jeans, leggings, and jeggings, should be avoided. Wear comfortable cotton clothes. Don't share sheets, towels, or clothing. Dust, wet mop, or vacuum the house, and clean with soap and detergent to minimise fungal spore load in the immediate area.
Medicover Hospitals
https://www.medicoverhospitals.in › diseases › fungal-inf...
Fungal Infection: Dos and Don'ts, Treatment, Causes & Risks
About featured snippets
Feedback
People also ask
Which food can reduce fungal infection?
Should I keep fungal infection dry or wet?


youtube.com/results?search_query=Fungal+infection+videos
Gmail
YouTube
Maps
DISEASE PREDICTIO...
YouTube
Fungal infection videos
Home
Shorts
Subscriptions
Library
History
Your videos
Watch later
Liked videos
Show more
Subscriptions
Sparx Motivation
Niraamay Wellne...
wonderful videos
GATE ACADEMY ...
Filters
Fungal Skin Infections Overview
JJ MedEd
Overview of Fungal Skin Infections | Tinea Infections
896K views • 4 years ago
JJ Medicine
Lesson on Tinea (Fungal) Skin Infections. Tinea infections are fungal infections of the skin caused by fungi of the genus ...
3:28
Tinea capitis is a fungal infection of the head and an easy way to remember this is capitis, "cap", so you can think of the cap or ...
CC
From health sources
FUNGAL INFECTION CAN BE TOTALLY CURED
JAYPEE
Fungal Infection can be totally cured under the treatment of an expert doctor
51K views • 2 years ago
Jaypee Hospital
Fungal infection is a common problem during summertime. It can be seen in any age group and in any part of the body. To avoid ...
Cleveland Clinic
How to Treat a Yeast Infection
708K views • 1 year ago
Cleveland Clinic
https://www.youtube.com/watch?v=qAjfOuTIWCI&pp=ygUXRnVuZ2FsIGluZmVjdGlvbiB2aW...


google.com/maps/search/diabetes+doctor+near+Solapur,+Maharashtra/@17.6638889,75.8869517,14z/data=!3m1!4b1?entry=ttu
Gmail
YouTube
Maps
DISEASE PREDICTIO...
diabetes doctor near Solapur, Maharas
Saved
Recents
Shrimati Bhuriben...
Dr Bhaskar Patil's Solapur Diabetes Care Center solapur
4.6 (66)
Hospital · rickshaw stop, Railway Line
Closed · Opens 10 am · 0217 231 6909
"God of diabetes patients... God bless you all"
Directions
DR.SACHIN MULAY DIABETES CARE CENTRE
4.9 (7)
Diabetes center · 2ND FLOOR, RAILWAY LINES ROHAN COMMERCIAL CENTRE
Open · Closes 6 pm · 0217 231 7546
"He is the Best doctor for Diabetic patients."
Directions
Dr Aseem Saifan's Lifecare Clinic-Diabetologist/Diabetes Specialist/Thyroid/Physician
5.0 (57)
Diabetologist · Plot No. 145 A, Jilla Parishad Chowk, Ruby Residency, sur no.6188/1b/c
Closed · Opens 1 pm · 090053
Website
Directions
Update results when map moves
Rating
Hours
All filters
Search this area
Solapur Bhuikot Fort
Siddheshwar temple
Zudio - Solapur, Maharashtra
CNS Hospital
Jay Info Tech
Jain Hospital - Advanced...
Dr. SANTOSH MOTE | Best... Temporarily closed
Joshi Clinic
Old Vidi Gharkul Solapur
KDMadur Onion merchant
Dr. Ravindra R. Gundeli
Pulgam Textiles Showroom
Walchand Institute Of Technology (WIT)
SAKHAR PETH
Dond Speciality Clinic
Manorama Heart, Diabetes, Thyroid...
Valsangkar Hospital | Best neurosurgeon in...
The Officer's Club
Dr. Smita Rahul Swami
Apna Bazar
HDFC Bank
Dy.RTO Solapur
Sai Super Market
Vijayraj Residency
Layers
Map data ©2023
India
Terms
Privacy
Send feedback
1 km


About
Disease Prediction through Symptom Analysis with Machine Learning
Exit
Disease Prediction Tab
In this tab, the user selects five health parameters (possibly symptoms or other indicators), and based on these inputs, the system predicts the most likely disease. It's not clear from the code how the prediction is made or what algorithms are used.
Test Tab
After the disease is predicted in the Disease Prediction Tab, this tab provides the names of tests that are typically associated with the predicted disease. These tests can help confirm or rule out the predicted disease. The tests are displayed in a list format.
Tips Tab
This tab provides tips related to the predicted disease. The tips can include suggestions for yoga exercises, dietary recommendations, and links to relevant YouTube videos. However, the code you provided doesn't contain the implementation of this tab, so it's not clear how the tips are generated or displayed.
Nearest Doctor Tab
This tab helps users find the nearest doctors or medical professionals based on their location or other search criteria. The code for this tab is not provided, so the implementation details are unknown.

Chapter 8

10. FUTURE WORK

- We will make an Android application for this disease prediction in which we willadd some more diseases.
- Also recommend some nearby hospitals according to the user's location anddisease. The future scope of the paper is the prediction of all diseases by

by using advanced techniques and the algorithm is less time complexity

Conclusion

- Various disease detected model has been developed using ML classification modeling techniques. This project predicts whether the patient has a disease or not by Extracting the patient's medical history from a dataset from a dataset such as a chest pain, blood pressure, etc.
- The algorithms used in building the given model are Logistic regression, Random forest classifier, KNN, etc.
  The accuracy of the kindly disease prediction model is 100% the accuracy of the diabetes prediction model is 80%.
- In conclusion, disease prediction has emerged as a valuable tool in healthcare, aiding in the early detection, prevention, and management of various illnesses. Through the use of advanced technologies, big data analytics, and machine learning algorithms, healthcare professionals and researchers can analyze vast amounts of data to identify patterns, risk factors, and indicators of diseases.
- Disease prediction models can utilize various data sources, including patient records, genetic information, lifestyle factors, environmental data, and social determinants of health. By integrating these diverse data sets, predictive models can generate accurate assessments of an individual's risk for developing specific diseases.
- The benefits of disease prediction are numerous. Early detection of diseases enables healthcare providers to initiate timely interventions and treatments, potentially preventing or minimizing the progression of illnesses. Predictive models also assist in resource allocation and healthcare planning, allowing for targeted interventions and allocation of resources to high-risk populations.
- Furthermore, disease prediction empowers individuals to take proactive measures to improve their health and reduce their risk of developing certain diseases. By identifying their personal risk factors, individuals can make informed decisions regarding lifestyle modifications, screening tests, and preventive measures.

Plagiarism Report :